Surrogate Genetic Programming: A Semantic Aware Evolutionary Search

Ahmed Kattan

AI Real-world Application Lab, UQU, Saudi Arabia

Yew-Soon Ong

Computer Science Department in Nanyang Technological University, Singapore

Abstract

Many semantic search based on Genetic Programming (GP) use a trial-and-error scheme to attain semantically diverse offspring in the evolutionary search. This results in significant impediments on the success of semantic-based GP in solving real world problems, due to the additional computational overheads incurred. This paper proposes a surrogate Genetic Programming (or sGP in short) to retain the appeal of semantic-based evolutionary search for handling challenging problems with enhanced efficiency. The proposed sGP divides the population into two parts (μ and λ) then it evolves μ percentage of the population using standard GP search operators, while the remaining λ percentage of the population are evolved with the aid of meta-models (or approximation models) that serve as surrogate to the original objective function evaluation (which is computationally intensive). In contrast to previous works, two forms of meta-models are introduced in this study to make the idea of using surrogate in GP search feasible and successful. The first denotes a "Semantic-model" for prototyping the semantic representation space of the GP trees (genotype/syntactic-space). The second is a "Fitness-model", which maps solutions in the semantic space to the objective or fitness space. By exploiting the two meta-models collectively in serving as a surrogate that replaces the original problem landscape of the GP search process, more cost-effective generation of offspring that guides the search in exploring regions where high quality solutions resides can then be attained. Experimental studies covering three separate GP domains, namely, 1) Symbolic regression, 2) Even n-parity bit, and 3) a real-world Time-series forecasting problem domain involving three datasets, demonstrate that sGP is capable of attaining reliable, high quality, and efficient performance under a limited computational budget. Results also showed that sGP outperformed the standard GP, GP based on random training-set technique, and GP based on conventional data-centric objectives as surrogate.

Keywords: Semantic Space, Surrogate Model, Semantic-model, Fitness-model, sGP.

Email addresses: Ajkattan@uqu.edu.sa (Ahmed Kattan), asysong@ntu.edu.sg (Yew-Soon Ong)

1. Introduction

Evolutionary algorithms (EAs) are approaches that take their inspirations from the principles of natural selection and survival of the fittest in the biological kingdom. Among the many variants of EAs, Genetic Programming (GP) is among one of those that have withstood the realms of time with success stories reported in a plethora of real-world applications. In particular, GP has been deemed as capable of providing transparency into how decisions or solutions are made. While a technique to evolve trees was suggested by Cramer in 1985 [6], the field of GP was founded by John Koza in 1992 [16]. GP is a powerful learning engine inspired by biology and natural evolution, for automatically generating working computer programs. Based on Darwins theory of evolution, computer programs are measured against the task they are intended to do and then receive scores accordingly. The programs with the highest scores are considered the fittest. The fittest programs are selected to join the evolutionary process via three standard genetic operators: crossover, mutation and reproduction. These operators aim to amend the programs structure and create new offspring, which will hopefully be better. Computer programs are treated as a collection of functions and inputs for these functions; which can be seen as their genetic material. In the standard representations of GP, programs are represented as trees where the functions appear in the internal nodes of the tree and the inputs for the functions appear on the leaves. This representation flexibility adds an extra advantage to the GP since it can solve complex problems and even comes up with solutions beyond human thoughts in some cases.

In GP, the search space can be viewed from multiple facets; A) Structural space or Syntactic space or Genotype in which most GP systems operate on, in attempting to evolve/alter the trees, via the genetic operators, hoping to find better ones that translate to desirable behaviours, B) Phenotype or Semantic space where GP systems try to alter the behaviours of the programs directly, and finally, C) Fitness space where individuals are evaluated into numerics that quantify the quality in relation to solving the given problem. In general, it is common for the semantic space of GP to be represented in the form of real number vectors and are defined by the outputs of GP trees when their instructions (or functions) are executed.

In traditional GP, although remarkable success on different real-world problems have been achieved (e.g., [36], [14]), existing systems operate fundamentally within the syntactic space, ignoring the semantic information of the candidate GP trees or programs. Advancing studies on semantic GP have highlighted the potentials of using the semantic information within the search. In particular, incorporations on semantic information of candidate solutions into the GP evolutionary process have recently been reported to generate significant enhancements in the search performances ([33], [4], [26]). Generally, the term semantic refers to the chromosomes or individuals' behaviour (for example, represented in the form of a real-valued vector), while syntactic refers to the structure of the evolved program. In most real-world applications, if not all, there is no obvious relationship between the syntactic space and semantic space. Thus, small changes in the shape of a tree can result in a significant departure in the

resultant behaviour and vice-versa, any small deviations requested in the program behaviour (or semantic) would need major variations in the tree structure. Due to the complex structure of GP in the syntactic space, i.e., tree-like structure representation, it is hard to identify a suitable syntactic distance measure that correlates well with the fitness landscape. In other words, it is hard to define a distance (or similarity measure) that quantifies the structural similarities between two trees and at the same time quantifies the similarities between their fitness values. Even if such distance measure is available, it is mostly problem-dependent. Generally, it is often deemed easier to determine the distance that correlates fitness landscape between individuals' semantics (where semantics represented as vectors of real numbers) than between individuals' syntactic (i.e., tree-like structure representation). This is because it is easier to use a generic distance metrics that applies across problem domains between vectors than trees. For these reasons, it is often argued that the semantic space is easier to search upon, than the syntactic space.

Despite the increasing research interests and potentials of semantic GP, one of the main criticism has been on the excessive slow nature of the approach, attributed not only to the natural mechanisms of evolution which involves an iterative process of candidate solutions that evolves across many generations, but more importantly, the trial-and-error scheme used to facilitate semantically diverse offspring in the search have led to significant increase in the computational resources needed before convergence to credible and reliable results can be attained. In this work, our interest is to retain the appeal of GP algorithms, especially semantic GP, that can handle challenging problems with high quality designs at enhanced computational efficiency.

We present a study on surrogate Genetic Programming or sGP in short. The core characteristics and motivations for proposing the sGP can be summarised as follows: 1) Present Semantic GPs heavily follow a trial-and-error scheme [25][34][11], which led to highly computationally intensive search. 2) It is desirable to conduct a search using Semantic GPs for high quality solutions under a limited computational budget, 3) It is non-trivial to map from syntactic to the semantic space [25]. Particularly, in contrast to previous studies on semantic based operators, where efforts have been placed on exploiting individuals' behaviour to maintain semantic diversity in the search population, our study on sGP in this paper differs in the use of meta-models for exploring the semantic space, so as to enhance search efficiency through eliminating the need to evaluate each computationally intensive candidate solutions completely.

The proposed model of sGP divides the population into two parts μ and λ then it evolves μ portion of the population using standard search operators and the original objective function, while the remaining λ population via the surrogate. On the λ portion of the population, sGP uses two forms of meta-models, namely the "semantic-model" and "fitness-model" as the surrogate that aids in the GP search. The semantic-model prototypes the semantic space of the GP trees (genotype/syntactic-space), while the fitness-model maps the semantic space to the fitness space (more details are given in Section 3). In order to construct the semantic space, it is generally necessary to evaluate the GP programs completely. But this is often deemed as computationally intensive and impractical for many real-world complex applications. By using the semantic-model, on the other hand, solution individuals only need to be partially evaluated and the complete behaviours of the GP trees (i.e., fitness value) is then predicted via the

fitness-model. The semantic and fitness-models thus serve as a partial replacement or surrogate of the computationally expensive objective function in semantic GP, leading to enhanced search efficiency. As will be shown in the results section, sGP is capable of achieving good solutions with smaller numbers of function evaluations.

In summary, the contributions of this paper is fourfold: 1) to the best of our knowledge, this is the first successful attempt to apply surrogate model to Semantic GP with tree-like representation, 2) we introduce the notion of semantic-model and fitness-model as surrogate, 3) the use of surrogate model to approximate the semantic search space allows generalisation to a metric space that is non problem-dependent, and last but not least 4) an efficient form of Semantic-aware GP is presented, labelled here as Surrogate GP (or sGP).

The remainder of this papers is organised as follows: Section 2 presents some related works, while Section 3 describes sGP in details, Section 4 presents the results of empirical studies on the proposed framework, and finally Section 5 gives some conclusive remarks and discusses potential directions of future work.

2. Related Work

In this section, the related works on semantic aware operator in GP and a brief review of surrogate modelling in evolutionary computation are presented.

2.1. Semantic Aware Operators

Beadle and Johnson proposed semantic driven crossover and mutation for boolean problems in [4] and [3]. The key idea is to allow search operators to alter the behaviour of offspring programs. To this end, the proposed operators transform the parents and offspring programs to a reduced ordered binary decision diagrams (ROBDDs) representation for the purpose of identifying the semantic equivalence between programs. Parent and offspring are considered semantically equivalent if and only if they have the same ROBDD. Operators that produce semantically equivalent offspring to their parents are then eliminated, thus reducing the number of calls to the objective function. Although, the idea of transforming individuals to ROBDD representation is an interesting way to reduce the computational costs of evaluating individuals, the approach can only apply to boolean problems.

In [33] Uy *et. al.* proposed a semantic aware crossover operator for real-valued symbolic regression problems. In their work, the authors proposed four different scenarios to apply the new operator. In the first scenario, the crossover is applied if the two sub-trees (to be exchanged in the crossover operator) are semantically inequivalent. The logic behind this concept is to maintain a certain level of semantic diversity in the population. In the second scenario, the crossover is applied if the two sub-trees (to be exchanged) are semantically equivalent. In the third scenario, the semantics of offspring are compared against their parents and inserted into the new generation if and only if they are not equivalent to their parents, otherwise the parents will survive to the next generation. The fourth scenario is the reverse of the third scenario, wherein offspring survive into the next generation if and only if they are semantically equivalent to their parents. Experimental results show that only the first suggested scenario led

to improved performance semantic GP. In [26], the same authors extended their work where the Semantic Aware Mutation (SAM) and Semantic Similarity Mutation (SSM) are proposed. In SAM, the semantic equivalence (or semantic distance) of the original sub-tree and the newly generated sub-tree (in the mutation operator) is determined by comparing the absolute difference of their behaviours on a set of random points in the domain. If the behaviours of the two sub-trees on the set is close enough (subjected to a parameter known as semantic sensitivity), they are designated as semantically equivalent. SSM compares the semantic similarity between sub-trees (the original and the newly generated) rather than complete semantic equivalence. Then, the semantic similarity of the two sub-trees is checked by comparing them to a set of random points in the domain. If the absolute difference between the behaviours of these sub-trees in the set fulfils some predefined threshold, they are considered as semantically similar, hence mutation proceeds. Otherwise, the system selects a new mutation point and randomly generates a new sub-tree. This loop iterates until it has reached the maximum number of trials defined. Recently, Moraglio *et. al.* [25] also proposed the Geometric Semantic GP (GSGP). The theory states that geometric crossover with regard to the metric d (metric d is any distance measure used by the representation), will result in offspring in the metric segment between parents. In geometric mutation with regard to the metric d the result offspring will fall within the ball of radius ϵ centred at the parent. The authors claim that semantic landscapes are always spherical, which help GP to identify superior solutions.

In spite of the increasing interest in semantic GP, it is worth noting that none of the previous works have considered the use of surrogate model to approximate the semantic space as means of speeding up the search on computationally expensive problems. In this paper we take semantic aware operators one further step and explore the potential benefits of semantic and fitness models as surrogate of the computational intensive objective function in GP. The results of this research is a new Surrogate GP that is capable of achieving good solutions with a smaller evaluation incurred.

2.2. Surrogate Modelling

Surrogate models (SMs) used in evolutionary frameworks, typically known as response surface models or meta-models, are approximation models that mimic the behaviour of the simulation model as closely as possible while being fast surrogates for time-consuming objective functions. In a nutshell, SMs work by running simulations at a set of points and fitting response surfaces to the resulting input-output data. To date, many data centric approximation methodologies used to construct surrogates. These include the polynomial regression, support vector machines, artificial neural networks, radial basis functions, Gaussian process [17, 35, 40, 30, 21] and surrogate ensembles [37, 10, 1, 39] are among the most commonly investigated [12, 19].

The general consensus on surrogate-assisted evolutionary frameworks is that the efficiency of the search process can be improved by replacing, as often as possible, calls to the costly objective functions, with surrogates that are deemed to be less costly to build and compute. In this manner, the overall computational burden of the evolutionary search can be greatly reduced, since the effort required to build the surrogates and to use them is much lower than those in the traditional approach that directly couples the evolutionary algorithm (EA) with the costly objective functions [8, 32, 27, 12, 31].

Early approaches have focused on building global surrogates [32] that attempt to model the complete problem fitness landscape. However, due to the effects of the curse of dimensionality [7], many have turned to local surrogate models [9, 28] or their synergies [39, 38] or ensembles. The use of domain knowledge including gradient information and physics-based models [15, 20] to improve the prediction accuracy of the surrogates have also been considered. The use of more than one surrogate, for example ensemble and smoothing surrogate models, and with preference for surrogates that generate search improvements over prediction accuracy was also considered in both single and multi-objective optimisation [18].

In principle, present surrogate models used are implicitly or explicitly spatial meta-models, as their predictions involve exploiting some assumed spatial smoothness or continuity between the values of the objective function at a query point whose value is unknown and has to be predicted based on the known solutions in the search space. This makes objective meta-models naturally suited to continuous function optimisation. As such, the plethora of research studies that incorporate SMs to speedup evolutionary search on computationally expensive problems were made on problems involving continuous or real-valued variables. Particularly, if the input variables are real-valued, the simple Euclidean distance can be readily used. When the defined distance-measure correlates well with the fitness landscape, the meta-model is generally capable of approximating the search space. However, if the distance-measure just quantifies structural differences between solutions without consideration of their fitness values, then using the surrogate can mislead the search process since they may not approximate the search space well.

To the best of our knowledge, no works in the literature have successfully defined surrogate models on complex representations, such as the GP tree-like representation, other than real-valued vectors. When dealing with complex representation like the GP tree-like representation, there is no general distance measure that correlates the fitness landscape for all problems, hence surrogate models cannot be easily used in GP search. Therefore, for search problems naturally based on structured representations, surrogate models can be used only after transforming the original representation to real-valued vector form. This introduces extra non-linearity in the target expensive objective function, results in making it harder to learn, and consequently requiring more expensive samples to approximate it well enough to locate its optimum. In [24], an attempt was presented to apply RBFN surrogate model on GP, but it was concluded that the results reported were not significantly different from those obtained by standard GP systems. In this paper we present a new framework that incorporates surrogate models to the GP tree-like representation successfully in Semantic GP. The proposed framework introduces the notion of semantic-model and fitness-model as the surrogate that maps the syntactic space to semantic space and then from the semantic space into fitness space, respectively. Further details are presented next.

3. sGP

As mentioned previously, sGP divides the population into two parts. We use the notation μ to denote the percentage of the population that will evolve using standard GP search (i.e., based on the original objective function evaluation) and λ to denote

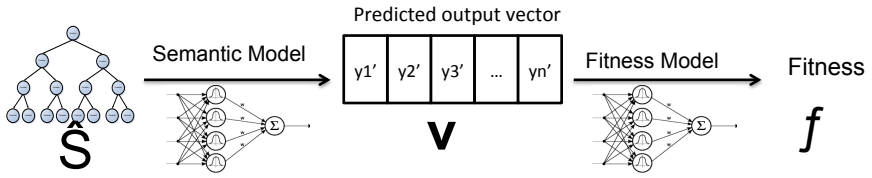

Figure 1: "Semantic model" serves to map a solution individual from the structural space S to semantic space V (i.e., $S \rightarrow V$ where it models the function $F(Tree_i, X_{sub}) = \mathbf{y^i_{sub}}$ to produce a prediction of the $Tree_i$'s semantic output based on the complete dataset X in the form $\hat{\mathbf{y}}_\mathbf{i} = \hat{\mathbf{F}}(\mathbf{Tree_i}, \mathbf{X})$) and then the "Fitness model" maps the solution individual in semantic space V to the fitness space F (i.e., $V \rightarrow S$ where it models the function $F(\hat{\mathbf{y}}_\mathbf{i}) = \hat{fitness}_i$).

the remaining percentage of the population that will evolve using the surrogate. To this end, in sGP, $\mu + \lambda = 1$.

Similar to standard GP systems, sGP begins with the initialisation of a random population and evaluates the μ part of the population using the original objective function $f(Tree_i, X), X = \{(\mathbf{x_j})|\mathbf{j} = \mathbf{1}, ..., \mathbf{n}\}$ to arrive at the corresponding exact fitness value $fitness_i$. Here, X is the set of training cases used to calculate the fitness. All non-duplicate individuals in the μ part of the population and their associated exact fitness values of $f(Tree_i, X)$ obtained during the search are archived in a centralised database D (we will explain the use of this database later in section 3.3).

3.1. Surrogate Building

In this work, we used Radial Basis Functions Network (RBFN) as the approximation methodology for surrogate modelling [5] due to its well-established effectiveness and ease of implementations. However, in principle, any surrogate model can be used in this framework. As mentioned previously, in the proposed sGP, two forms of surrogate within the (λ) sGP evolution. We will call the two surrogates, 1) semantic RBFN model, and 2) fitness RBFN model (both act as replacement of the original computationally expensive problem space), as depicted in Figures 1 and 2. Naturally, the (λ) sGP portion of the evolutionary search is computationally cheaper than the original problem since it is guided by computationally cheap surrogates instead of the problem's original fitness measure.

As illustrated in Figure 1, if the structural space, semantic space and fitness space is S, V and F, respectively, the semantic-model approximates the $Tree_i$ input-output relationships or the semantics of the problem by mapping $S \rightarrow V$. The fitness-model then maps $V \rightarrow F$. Both approximation models are thus used collectively to approximate the original syntactic space (which has a non-continuous variable space representation) to the real-valued semantic space which is continuous, such that prediction of fitness values for GP tree-like representation can be achieved at reduced computational cost.

The next sub-section will explain the representation of the RBFN surrogates used in this model.

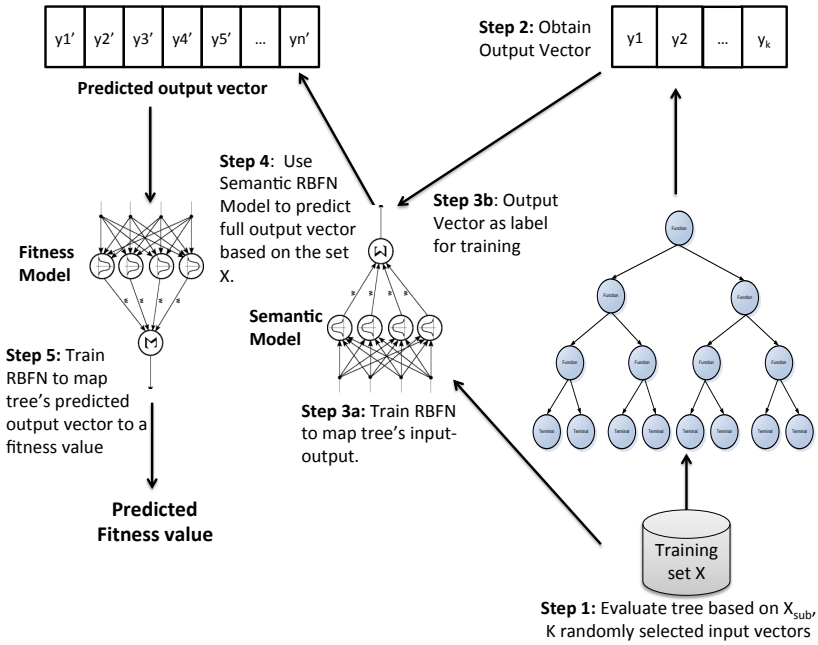

Figure 2: Workflow outline of fitness evaluation in the portion of (λ) sGP.

3.1.1. RBFN Training

The evaluation of the functions in $y = f(Tree_i, X)$ may be time consuming. Thus, instead of evaluating these functions, we want to replace it with a surrogate model that approximates $\hat{y}_i = \hat{F}(Tree_i, X)$.

RBFNs can be seen as a variant of artificial neural networks that uses radial basis functions as activation functions [5]. They have been used in function approximation, time series prediction, and control [5]. A radial basis function (RBF) is a real-valued function $\phi : \mathbf{R}^n \rightarrow \mathbf{R}$ whose value depends only on the distance from some point c, called a *center*, so that

$$\phi(\mathbf{x}) = \phi(\|\mathbf{x_q} - \mathbf{c}\|)$$

The point c is a parameter of the function and the point x_q is the query point to be estimated. The norm is usually Euclidean, so $\|\mathbf{x}-\mathbf{c}\|$ is the Euclidean distance between c and x. The most common used type of radial basis functions is the Gaussian functions as in:

$$\phi(x) = \exp(-\beta\|\mathbf{x} - \mathbf{c}\|^2)$$

where $\beta > 0$ is the width parameter. Radial basis functions are typically used to build function approximations as in Equation 1.

$$\hat{y(\mathbf{x})} = w_0 + \sum_{i=1}^{N} w_i \, \phi(\|\mathbf{x} - \mathbf{c}_i\|) \tag{1}$$

Thus, $\hat{y(\mathbf{x})}$ is used to approximate $f(Tree_i, X)$. The approximating function $\hat{y(\mathbf{x})}$ is represented as a sum of N radial basis functions, each associated with a different center c_i, a width β_i, and different weight w_i, plus a bias term w_0. In principle, any continuous function can be approximated with arbitrary accuracy by a sum of this form, if a sufficiently large number N of radial basis functions is used. The bias w_0 can be set to the mean of the values of the known data-points from the training set that used to train the RBFN.

Training the RBFN requires to find three parameters: *i)* the centres $\mathbf{c}_i$, *ii)* the values of w_i in such a way that the predictions on the training set minimises the errors and *iii)* the RBF width parameters β_i.

The centers are chosen to coincide with the known data-points and evaluated with the real fitness function. The β value is fixed for all N linear RBFs (global). In this paper, we set β for each RBFs as $1/MD^2$ where MD is the max pairwise Euclidean distance between the query data-point and all n in the training set.The value of β controls the radius of each RBF to spread on the space to cover all other centres so that each known function value at a center can potentially contribute significantly to the prediction of the function value of any point in space, and not only locally to function values of points near the given center.

Finally, the weights vector is calculated by solving the system of N simultaneous linear equations in w_i obtained by requiring that the unknown function interpolates exactly the known data-points

$$y(\mathbf{x}_i) = b_i,\ i = 1 \dots N$$

By setting

$$g_{ij} = \phi(||\mathbf{x}_j - \mathbf{x}_i||),$$

the system can be written in matrix form as $\mathbf{Gw} = \mathbf{b}$ where $\mathbf{b}$ is a vector of the true fitness values of the data-points that have been used to train the surrogate. The matrix $\mathbf{G}$ is non-singular if the points $\mathbf{x}_i$ are distinct and the family of functions ϕ is positive definite (which is the case for Gaussian functions). Thus solving $\mathbf{w} = \mathbf{G}^{-1}\mathbf{b}$ gives the weights w.

The value of the bias term w_0 in Equation 1 is set to the mean value of the known data-points, i.e., the mean of vector $\mathbf{b}$. So the predicted function value of a point which is out of the influence of all centres, is by default set to the average of their function values.

The inputs that have been given to the fitness RBFN model and semantic RBFN model will be explained in details in sections 3.2 and 3.3, respectively.

3.2. Semantic-Model

Generally, the process of obtaining the semantic representation for trees in GP population requires full evaluation for each tree using the fitness measure (which could be computationally expensive). However, in sGP we only need to partially evaluate each GP tree using $k = |X_{sub}|$, as opposed to a fully complete evaluation of the trees, i.e., since $k << n$, where $n = |X|$. Thus, significant computational cost savings can be attained.

The main task of the semantic-model is to predict the semantics of GP trees, represented in the form of real-valued vectors. The semantic RBFN model is designed to yield insights into the functional relationship between the syntactic input $F(Tree_i, X_{sub})$ and semantic output vector $\mathbf{y^i_{sub}}$ (hence established in this paper as a semantic aware evolutionary search) by prototyping the function $F(Tree_i, X_{sub}) = \mathbf{y^i_{sub}}$ such that $X_{sub} \subset X$ where $|X_{sub}| << |X|$. In other words, X_{sub} is subset of the training set X, randomly selected. For each individual in the λ part of the population, an approximation or prediction of the $Tree_i$ semantic output based on the complete dataset X is made using the semantic-model. This can be represented as $\hat{\mathbf{y}}_\mathbf{i} = \hat{\mathbf{F}}(\mathbf{Tree_i}, \mathbf{X})$ such that $\mathbf{y_i} = \hat{\mathbf{y}}_\mathbf{i} + \epsilon$. Next, in taking $\hat{\mathbf{y}}_\mathbf{i}$ as input of the fitness-model (details in section 3.3), we approximate the original fitness function $f(Tree_i, X)$, where $\hat{\mathbf{y}}_\mathbf{i}$ denotes the output of the semantic-model and $\hat{fitness}_i$ is the predicted fitness value given $\hat{\mathbf{y}}_\mathbf{i}$, which is the inferred behaviour output of $Tree_i$ [1].

One can argue that the cost of using the RBFN surrogate requires computational cost itself and thus the whole process could end without any reduction in computational costs. However, remember that our main assumption is that the proposed model works with computationally expensive problems (e.g., modelling of mechanical engines or simulation of natural phenomena) where fitness measure can be time consuming and in these cases the cost of using the RBFN can be negligible. To formalise the computational saving of semantic calculations in sGP, if the size of the training set is n and the number of solution trees explored by the (λ) sGP is E, (λ) sGP would have performed $\frac{E \times k}{n}$ evaluations based on the exact fitness measure (which we assume it is a computationally expensive process) to construct the semantic model. In contrast, existing semantic aware operators fully evaluate each GP tree completely in the population. To this end, if E is the number of exact fitness evaluations performed by standard semantic aware operators to obtain the tree semantics, the computational cost difference between the (λ) sGP and standard semantic aware operators is $E : \frac{E \times k}{n}$, where $n >> k$ and $E > 0$. Thus, the evaluation of sGP in obtaining the tree semantics can be achieved at lower computational costs.

Once the semantic-surrogate (based on RBFN) is trained using k points (which is relatively inexpensive comparing to evaluating a tree on n points) it becomes possible to predict the semantics of given trees. Hence, we reduce the full evaluation of each tree (on n number of training samples) required by standard semantic operators to only k training samples. A disadvantage of this model is the low number of training samples used to train the RBFN (i.e., low k) may lead to questioning the ability of sGP to perform adequately on problems with noise, missing samples, or outliers. However, in preliminary experiments of sGP we found that when $k = 40\%$ of the training set (randomly selected) a reasonable performance is attained.

One can argue that there is no need to predict the complete semantics (denoted by real-valued output vectors of dimension R) of trees and search semantic space altogether, for computationally expensive problems, since the fitness value of each tree can be directly inferred from the partial evaluation (i.e., those k training samples). Al-

[1] In this paper, the 'semantic' space is spanned by real-valued vectors of dimension R. Thus, the term semantic refers to the GP tree output behaviour denoted by real-valued vectors.

though this argument may sound logical, we demonstrated that evaluating GP trees based on a small training set make GP system more likely to converge towards false optimum and leads the GP system to overfit the training data, thus losing generalisation capabilities. Moreover, we shall show in the experiments section, the results differ between two versions of sGP. In the first version, we switched off the fitness-model and guide the (λ) sGP search using only the semantic-model. In the second version, we enable the fitness-model. Results obtained (as will be shown later in Section 4) demonstrated that the collective use of both models as the surrogate led to superior GP performance.

3.3. Fitness Model

As illustrated in Figure 2, the semantic-model passes the predicted behaviour output vector $\hat{\mathbf{y}}$ of each tree in the (λ) sGP population, to the fitness-model.

Taking the predicted behaviour output vector as input, the fitness-model serves to map the trees' predicted output vectors (as produced by the semantic-model) to their fitness values. This can be formalised as follows. Let the fitness-model be denoted as $F(\hat{\mathbf{y}}_\mathbf{i}) = \hat{fitness}_i$, where $\hat{fitness}_i$ denotes a prediction of the $Tree_i$'s fitness value.

In order to train the fitness-model, we used individuals in the μ sGP. As mentioned previously, we store semantics of the elite non-duplicate μ individual solutions and their associated exact fitness values of $f(Tree_i, X)$ obtained during the search in a centralised database D. Fitness-model uses the natural distance of the underlying output vectors' representation to map trees' semantics to their fitness values. Hence, if the trees' output vectors are real-valued, simple Euclidean or Manhattan distance can be used. If the trees' output vectors are binary values then Hamming or Edit distance can be used.

The next section will present the experimental settings and results.

4. Experiments

4.1. Experimental Settings

The aim of the reported experiments is to show that sGP is able utilises the surrogate model to explore the search space effectively under limited small number of evaluations (assuming that the given problem is computationally expensive) and achieve superior solutions in comparison to other versions of GP systems when given exactly the same number of evaluations. Experiments are made to cover three different classic GP problems; 1) six different symbolic regression problems to test sGP on continuous problems space, 2) four different even n-parity bit problems to test the model on discrete problems space, and finally 3) a real-world application domain involving three different time-series forecasting problems to further validate the sGP.

We compared the sGP against the 1) standard GP (GP) system to check whether the idea of surrogate approximation for semantic space is useful, 2) standard GP with random training technique (GP-RT) to allow GP to evolve generalised solutions using a small training set, 3) GP based on a standard global surrogate model. The global surrogate directly searches the syntactic space of the programs using the Structural Hamming Distance [23] (GP-Surrogate(SHD)). We included GP-Surrogate(SHD) in order

to compare surrogate approximation based on semantic differences against surrogate approximation based on syntactic differences.

Table 1: sGP Settings used in the experiments.

Parameter	*sGP*
μ	0.6
λ	0.4
Mutation of μ pop	30%
Crossover of μ pop	70%
Tournament size	2
Population Size	10
Generations	10
λ *sGP*	
Mutation	30%
Crossover	70%
Tournament size	2
Population Size	20
Generations	20
Training samples for semantic-model k	4

We treated all test cases as computationally expensive problem representatives. Thus, we limited the evaluation budget in all systems in the comparison to only 100 objective function evaluations. The term 'budget' used here to indicate the total number of evaluations used by the GP systems in the comparison to explore the search space. Hence, for all systems included in the comparison, we set the population size and maximum number of generations 10 (i.e., 100 evaluations only to find the best possible solution). Also, without loss of generality, we considered all problems as minimisation problems (i.e., solutions with lower fitness values are better). Table 1 elucidates the settings of sGP. To allow fair comparisons, we gave all algorithms in the comparison exactly the same number of fitness evolutions. For the GP and GP-RT systems, 70% sub-tree crossover and 30% sub-tree mutation was used as search operators. In the GP-RT, a sub set of the training cases (of size 4), randomly selected, from the training set and changed every generation. GP-Surrogate(SHD) is treated as sGP and received the same settings, however the difference is that sGP uses two specialised surrogate models to the semantic space (see Section 3) while GP-Surrogate(SHD) uses only global RBFN based on structural Hamming distance. For the μ and λ parameters, we tried different values and found that 40%-60% is the best ratio to obtain the best results (more details on the justification these settings in Section 4.2.6). Finally, all GP systems in the comparison used rump half-and-half [29] for tree initialisation.

Table 2: Symbolic Regression Problems used in the experiments.

Function	Type
$F1 = x^3 + x^2 + 5x$	Polynomial
$F2 = \frac{x^6}{x^3+x^2+1}$	Polynomial
$F3 = \frac{x}{1-log(x^2+x+1)}$	Logarithmic
$F4 = Sin(x^2)$	Trigonometric
$F5 = 5\sqrt{\lvert x \rvert}$	Square-root
$F6 = 100 + log(x^2) + 5\sqrt{\lvert x \rvert}$	Square-root + Logarithmic
$F7 = 2 \times (tan(x) \times cos(x))$	Trigonometric

4.2. Results

4.2.1. Symbolic Regression

To study the performances of the different GP systems on continuous problem domains, we used six real-valued symbolic regression problems. These problems covered different types of functions: polynomial functions; trigonometric, logarithmic and square-root functions; and complex functions (logarithm plus square-root). Table 2 illustrates all the test functions used in the experimental study. For all functions, we randomly sampled 100 points from the interval $[-5, 5]$ and used them as a training-set, while using another 150 points as a testing-set to validate the ability of the systems to interpolate new data-points from the training-set. Both training and testing sets are disjointed. In addition, we randomly sample 50 new points from the interval $[-10, 10]$ to validate the systems' extrapolation ability. The average of absolute errors was used as a fitness measure. The statistical results (mean of the best-evolved solutions, median, standard deviation and best-solution) of the GP systems across 20 independent runs per test function are reported in Table 3. All GP systems have used $\{+, -, *, /\}$ as the function-set while $\{x, constant1 - 6\}$ as the terminal-set.

In addition, to verify the significance of the results differ between any two algorithms, we conducted the Kolmogorov-Smirnov two-sample statistical significance test on the results produced by the best-evolved solution of each system in comparison to sGP and reported the P value in the last column. The P value is configured with a 5% significance level. As seen in table, sGP achieved superior means on all six regression problems (as highlighted in boldface) with marginal differences ranging from 3% to 94%. The low means attained by sGP are also confirmed by the low medians in all cases. For the best-evolved solution, sGP managed to evolve higher quality solutions in five out of the six regression problems with marginal differences ranging from 0.1% to 88%. Now, on the unseen testing data (illustrated in Table 4), the proposed framework is also shown to display superior means and medians in five out of the six cases understudy, with marginal differences ranging from 2% to 94%. As for the best-evolved solutions, which test the generalisation abilities the proposed sGP also fares best in five out of the six cases, with marginal differences ranging from 1% to 93%. One may expect that GP-RT to evolve solutions that generalise well on the unseen testing set. On the contrary, as a result of the limited computational budget available (small number of evaluations), GP-RT did not have sufficient computational resources to expose all individuals to the entire training examples and was unable to evolve towards generalised

solutions. This suggests that GP with random training does not pose as a viable solution for computationally expensive problems where computational budget is limited. The P value is at the standard 5% significance level in most cases in the training and testing sets. Finally, Table 5 reports the extrapolation performance where sGP is noted to have emerged as superior in only three out of the six cases. Results show that sGP can not extrapolate well from the training set in comparison to its competitors.

Table 3: Symbolic Regression: Summary results of 20 independent runs on (Training set).

Algorithm	Mean	Median	Std	Best	P-value
		F1			
GP	6.95	6.94	5.71	9.6E-06	7.2E-04
sGP	**1.12**	**1.1E-05**	3.52	**7.9E-06**	N/A
GP-Surr(SHD)	5.67	2.24	6.36	8.8E-06	0.0232
GP-RT	18.66	23.80	11.59	9.1E-06	8.E-06
		F2			
GP	12.42	11.62	6.41	3.44	4.15E-05
sGP	**5.56**	**3.83**	4.65	**2.57**	N/A
GP-Surr(SHD)	13.37	13.75	7.50	2.77	1.83E-04
GP-RT	35.57	35.03	7.70	21.71	5.55E-10
		F3			
GP	4.72	3.04	7.74	**1.23**	0.2753
sGP	**4.48**	**2.72**	7.84	1.24	N/A
GP-Surr(SHD)	4.74	3.12	7.71	1.27	0.771
GP-RT	7.04	4.49	8.76	2.27	7.25E-04
		F3			
GP	0.53	0.54	0.04	0.47	0.4973
sGP	**0.52**	**0.52**	0.04	**0.44**	N/A
GP-Surr(SHD)	0.54	0.54	0.03	0.47	0.4973
GP-RT	2.74	0.81	3.65	0.52	4.15E-05
		F4			
GP	1.67	1.81	0.46	0.87	8.42E-06
sGP	**0.84**	**0.85**	0.23	**0.38**	N/A
GP-Surr(SHD)	1.91	2.01	0.47	0.85	2.49E-07
GP-RT	3.03	2.87	1.01	0.81	4.74E-09
		F6			
GP	3.94	3.99	0.87	1.79	0.0026
sGP	**2.72**	**2.84**	0.93	**1.04**	N/A
GP-Surr(SHD)	3.87	3.86	0.94	1.40	7.25E-04
GP-RT	32.21	5.26	111.43	3.03	1.53E-06
		F7			
GP	2832.48	447.07	4991.47	**27.03**	1.0
sGP	**2831.54**	**446.96**	4991.62	27.14	N/A
GP-Surr(SHD)	2832.48	447.17	4991.52	27.17	0.0026
GP-RT	2 832.90	447.17	4991.48	28.02	0.0014

*Bold numbers are the lowest.

* GP-Surr(SHD) is GP-Surrogate(SHD).

Table 4: Symbolic Regression: Summary results of 20 independent runs on (Testing set).

Algorithm	Mean	Median	Std	Best	P value
		F1			
GP	8.21	8.79	6.73	9.8E-06	7.25E-04
sGP	**1.10**	**1.3E-05**	3.37	**8.8E-06**	N/A
GP-Surr(SHD)	6.21	2.18	7.24	9.4E-06	0.0232
GP-RT	19.57	24.70	12.14	1E-05	8.42E-06
		F2			
GP	20.59	15.30	20.67	2.78	0.0082
sGP	**13.45**	**6.37**	20.18	**1.92**	N/A
GP-Surr(SHD)	21.13	14.90	22.22	2.46	0.02
GP-RT	45.33	37.96	22.89	26.30	3.63E-08
		F3			
GP	9.09	3.03	22.59	**1.58**	0.4973
sGP	**8.89**	**2.71**	22.60	1.82	N/A
GP-Surr(SHD)	9.09	3.11	22.58	1.50	0.77
GP-RT	10.42	4.48	22.49	2.70	0.01
		F4			
GP	**0.58**	**0.57**	0.05	0.50	0.0591
sGP	0.68	0.59	0.23	**0.47**	N/A
GP-Surr(SHD)	0.91	0.58	1.36	0.48	0.77
GP-RT	2.19	0.88	2.71	0.54	0.02
		F5			
GP	1.71	1.82	0.52	0.89	7.25E-04
sGP	**0.99**	**0.89**	0.40	**0.24**	N/A
GP-Surr(SHD)	1.99	2.05	0.49	1.06	1.53E-06
GP-RT	3.16	2.91	1.10	0.89	4.74E-09
		F6			
GP	9.39	4.37	23.22	1.16	0.0082
sGP	**5.90**	**3.13**	7.91	**0.97**	N/A
GP-Surr(SHD)	4.15	3.91	1.28	2.50	0.06
GP-RT	16.98	5.15	41.63	3.38	7.25E-04
		F7			
GP	3.9E+04	**214.23**	1.6E+05	22.88	0.0591
sGP	**2.5E+07**	1226.25	1.0E+08	**22.87**	N/A
GP-Surr(SHD)	3.9E+04	231.22	1.6E+05	23.01	0.2753
GP-RT	3.9E+04	222.54	1.6E+05	22.88	0.1349

*Bold numbers are the lowest.

* GP-Surr(SHD) is GP-Surrogate(SHD).

Table 5: Symbolic Regression: Extrapolation performance of best solution across 20 independent runs.

Test Function	Mean
F1	
GP	84.28
sGP	**2.54**
GP-Surrogate(SHD)	61.77
GP-RT	174.99
F2	
GP	84.18
sGP	**11.66**
GP-Surrogate(SHD)	102.20
GP-RT	254.66
F3	
GP	**3.18**
sGP	4.94
GP-Surrogate(SHD)	3.38
GP-RT	5.09
F4	
GP	0.82
sGP	0.94
GP-Surrogate(SHD)	**0.71**
GP-RT	10.04
F5	
GP	5.11
sGP	**4.32**
GP-Surrogate(SHD)	5.14
GP-RT	6.32
F6	
GP	6.64
sGP	6.35
GP-Surrogate(SHD)	**4.49**
GP-RT	5.77
F7	
GP	3453.682
sGP	3468.688
GP-Surrogate(SHD)	**3453.12**
GP-RT	3454.74

*Bold numbers are the lowest.

4.2.2. Even n-Parity Bit

The even n-parity problem involves the function-set {AND, OR, NOT} and terminal-set $\{X_0, X_1, X_2, ..., X_n\}$. The objective is to return a logic $'1'$ when an even number of inputs $(X_0 - X_n)$ is present. We studied the sGP on the n-parity problem for $n = 34, 5, 6, 7$ and 10 variables. The fitness function is the percentage of the wrong outputs (i.e., minimisation problem). This problem set has been selected to validate the sGP under discrete problem domain. Little success on the use of surrogate models to approximate discrete landscape have been reported to date (e.g., see [13]). sGP's remarkable results are observed on the discrete problem. Table 6 reports the results of 20 independent runs for each GP system on the range of n-parity problem. On the 3-parity the problem is too easy where all algorithms have found a global optimum solution (with fitness value equals 0) except GP-Surrogate(SHD). On 4-parity, the problem is still simple to solve, thus sGP and GP-Surrogate(SHD) converged to the same statistical mean and median as well as the best-evolved solutions with similar fitness values.

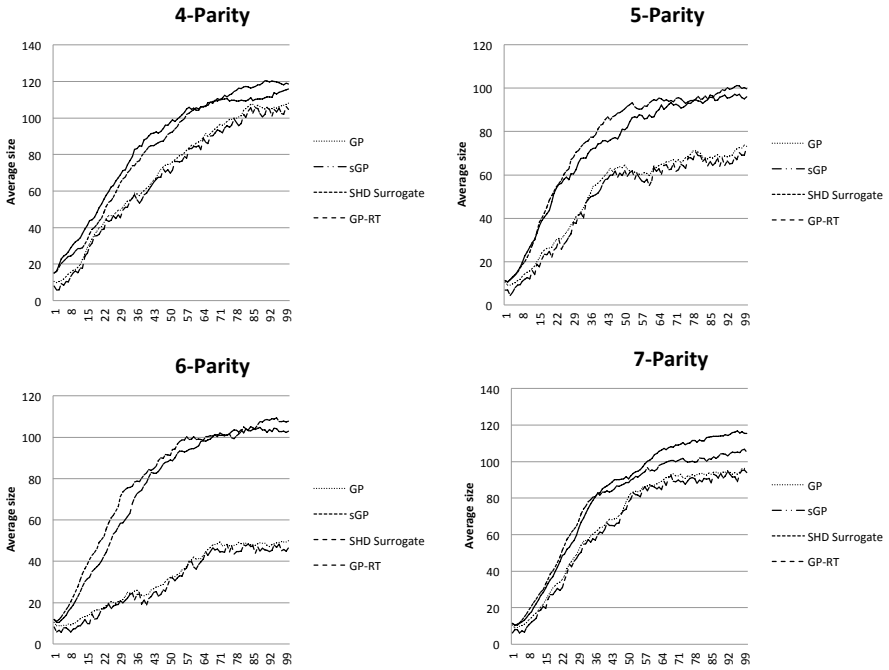

Figure 3: Population average trees' size (for n-Parity bit problems) graph for all GP systems in comparison (averaged over 20 independent runs).

On the larger $5, 6, 7$ and 10 parity problem statistical performance differences are more evident between the various methods, where sGP is noted to emerged as superior in the statistical mean and median and best-evolved solutions attained verified by the P reported in the last column in Table 6.

As shown in Figure 3, sGP and GP-Surrogate(SHD) tends to bias the search towards large trees while GP and GP-RT biases towards smaller trees. It is likely that in both sGP and GP-Surrogate(SHD) the used surrogate favours the exchange of large blocks between individual trees so as to alter the trees' semantics.

Table 6: Even n-Parity: Summary results of 20 independent runs.

Algorithm	Mean	Median	Std	Best	P value
3 variables					
GP	**31.25**	**0.00**	**37.50**	10.08	0.9655
sGP	33.75	**0.00**	**37.50**	8.93	N/A
GP-Surr(SHD)	33.75	25.00	**37.50**	5.73	0.9999
GP-RT	47.50	**0.00**	50.00	10.90	4.7406e-09
4 variables					
GP	36.25	**37.50**	2.64	**31.25**	0.324
sGP	**34.38**	**37.50**	3.29	**31.25**	N/A
GP-Surr(SHD)	**34.38**	**37.50**	3.29	**31.25**	1
GP-RT	48.75	50	2.64	43.75	1.28E-09
5 variables					
GP	46.25	**46.87**	1.32	43.75	0.1764
sGP	**45.00**	**46.87**	2.19	**40.63**	N/A
GP-Surr(SHD)	46.56	**46.87**	0.99	43.75	0.01
GP-RT	50.00	50	0.00	50.00	1.31E-10
6 variables					
GP	49.69	50.00	0.66	48.44	9.93E-04
sGP	**47.97**	**47.65**	1.29	**46.88**	N/A
GP-Surr(SHD)	49.06	50.00	0.81	48.44	0.00
GP-RT	50.00	50.00	0.00	50.00	3.98E-08
7 variables					
GP	48.98	49.21	0.38	48.44	0.045
sGP	**48.67**	**48.43**	0.53	**47.66**	N/A
GP-Surr(SHD)	49.14	49.22	0.25	48.44	4.05E-04
GP-RT	50.00	50.00	0.00	50.00	9.60E-10
10 variables					
GP	50	50	0	50	0.0486
sGP	**49.9**	50	0.05	**49.9**	N/A
GP-Surr(SHD)	50	50	0	50	0.0486
GP-RT	50	50	0	50	0.0486

*Bold numbers are the lowest.

4.2.3. Time-Series Forecasting

In this subsection, we shall next study the efficacy of sGP on a real world application, particularly pertaining to a Mackey-Glass time-series and two other financial time-series predictions.

The Mackey-Glass time-series was introduced by Mackey and Glass in [22]. It has been widely used as a benchmark for the generalisation ability of a variety of machine learning methods. The Mackey-Glass equation is:

$$x(t+1) = x(t) - bx(t) + a\frac{x(t-\tau)}{1+x(t-\tau)^{10}}$$

where $a = 0.2, b = 0.1$ and $\tau = 5$, the time series is aperiodic, non-convergent, and completely chaotic. Each GP system received 100 samples as a training set and another 150 samples as the testing set from the interval $[-100, 100]$. The function-set and terminal-set are defined as $\{+, -, *, /, sqrt, sin, cos, pow, log\}$ and $\{x, constant1 - 100\}$, respectively. For the financial time-series, we used the *Apple's* and *Boing's* weekly stock prices from 2002-2012 (freely available from Yahoo Finance service). The average of absolute errors was used as a fitness measure. Table 7 summarises the statistical results of the 20 independent runs. As can be seen, sGP

outperformed all counterpart systems on all three time-series forecasting problems in terms of mean, median and in 2 out of 3 cases in terms of best. sGP lost the comparison against GP with very small margins. It is fair to report that none of the GP systems understudy managed to accurately fit the training time-series. This is because the function-set given to the GP systems in the comparison is not enough to construct functions that fit the stochastic nature of the test time-series. Thus, all GP systems evolved functions that at best draw a smooth line through the target time-series.

Table 7: Time-Series Forecasting: Summary results of 20 independent runs.

Algorithm	Mean	Median	Std	Best
Mackey-Glass time-series				
GP	38.27	41.31	5.40	31.01
sGP	**28.95**	**28.48**	2.87	**24.87**
GP-Surr(SHD)	39.02	41.05	5.01	31.79
GP-RT	39.25	42.87	5.84	30.91
Boeing time-series				
GP	160.29	160.30	0.05	160.20
sGP	**159.48**	**159.77**	0.88	**157.87**
GP-Surr(SHD)	160.10	160.28	0.51	158.65
GP-RT	161.03	160.36	2.12	160.30
Apple time-series				
GP	190.30	190.33	0.40	**189.72**
sGP	**189.95**	**189.90**	0.10	189.75
GP-Surr(SHD)	190.20	190.17	0.35	189.87
GP-RT	203.79	199.74	12.02	199.04

*Bold numbers are the lowest.

4.2.4. sGP with Fitness-Model vs. sGP without Fitness-Model

As mentioned previously, sGP uses a semantic-surrogate model to predict the trees' output vectors (which we refer to as semantics in this paper) and a fitness-model to search the underlying semantic space of the programs. Of course, it is possible to calculate the fitness value of any tree once the semantic-surrogate predicts the output vector, which would enable one to avoid using the fitness-model altogether. However, we argue that semantic space is easier to search and thus the use of the fitness-model has direct effects on the performance. In this section, we test two versions of sGP to test our hypothesis. In the first version, we removed the fitness-model and calculated the fitness values of λ part of the population using the predicted output vectors produced by the semantic-surrogate alone. In the second version, we included the fitness-model. For the purpose of this study we used function $F6$ (see table 2) as the test bed. Each sGP version (i.e., one with fitness-model and one without fitness-model) tried to solve the given problem using 10 individuals in 100 generations. We compared the two versions with 20 independent runs. Figure 4 shows the best-in-generation graph (averaged over 20 independent runs). It is clear that sGP with a fitness-model (i.e., the second version) achieved improved solutions over the sGP without a fitness-model (i.e., the first version). The mean fitness (on training set) of best-solutions across the 20 runs for the first version is 139.15, while for the second version is 82.81. The fitness of the best-evolved solution across the whole 20 runs for the first version is 38.93 while

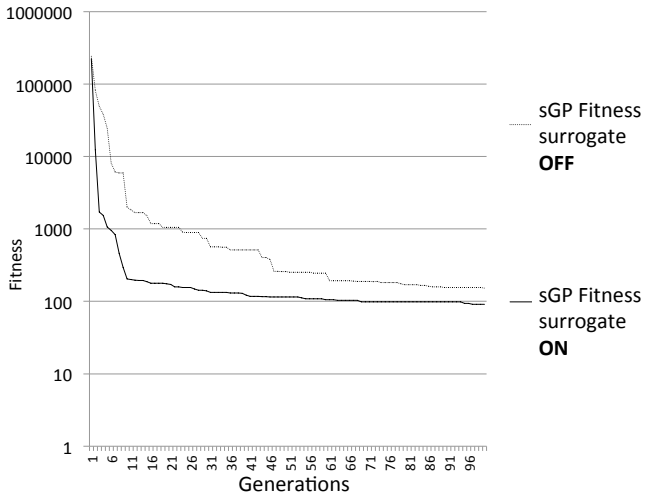

Figure 4: Performance differences between sGP with and without fitness-model model.

for the second version is 14.92. In addition, to verify the significance of the resulting differences between the two versions, we further conducted the Kolmogorov-Smirnov two-sample statistical significance test based on the best-evolved solution produced by the two versions across the 20 run. A P value of 0.0232 at 5% significance level is obtained.

It is also notable that the results differ greatly at early stages of the evolution, while in later generations, both version converge competitively. This shows that the proposed sGP is suitable for solving computationally expensive problems since it is able to identify high quality solutions rapidly and does not exhibit any degradation in performance at later stages of the search.

In our experiments, sGP is noted to have incurred a lower execution time than the standard GP system. However, it has an execution time that is close to GP-RT and GP-Surr (SHD). This is as expected since both GP-RT and GP-Surr (SHD) evaluate the trees only partially to improve computational efficiency. Nevertheless, in contrast to GP-RT which randomly selected the training sub-set, GP-Surr (SHD) performs a more intelligent means to improve computational efficiency via the use of global surrogate.

4.2.5. sGP in comparison against state-of-the-art regression models

As mentioned previously, the aim of the experiments is to demonstrate that sGP is capable of discovering optimum solutions under limited number of evaluations, assuming the given problem is computationally expensive where the user does not have the luxury of performing extensive search. However, in order to evaluate the behaviour of sGP under different circumstances, in this section, we will explore sGP performance when the number of evaluations is high against the other GP systems, thus, relax the search and allow evolution time to converge. For the purpose of this analysis, we used function $F4$ as a test bed. We used the same settings described in Section 4.1, however,

the number of evaluations has been increased to $10,000$ where the population size is 100 and number of generations is 100. All systems tried to approximate the function in 20 different runs. In addition, we included in the comparison Kriging, RBFN, and Linear and Polynomial Regression [2]. Table 8 shows the results of the comparison. Note that these algorithms are deterministic and therefore we applied them only once to solve the problem. sGP outperform all its comparators in terms of mean, median and best solutions. Interestingly, performance of GP-RT is competitive which confirm our assumption that GP-RT requires sometime to converge and it is not good when the search runs for small numbers of evaluations.

Table 8: Symbolic Regression: Summary results of 20 independent runs on (Testing set) in comparison against state-of-the-art regression models.

Algorithm	Mean	Median	Std	Best
F4				
GP	3.07	2.28	3.33	0.66
sGP	**0.63**	**0.62**	0.01	**0.62**
GP-Surr(SHD)	0.97	0.63	0.83	0.64
GP-RT	24.22	0.63	62.41	0.63
RBFN				26.23
Kriging				192.74
Linear Regression				1.20
Polynomial Regression				0.97

*Bold numbers are the lowest.

4.2.6. Sensitivity Analysis for μ and λ

As mentioned previously, we set parameters μ and λ based on trail-and-error in preliminary experiments. In this section, we will show the behaviour of sGP under different values for λ . We have conducted a sensitivity analysis for the parameter λ. For this purpose, we used sGP to solve four symbolic regression problems, namely F1-F4 (see Table 2) under different values of λ. As illustrated in Figure 5, sGP achieved the best results on problems F1,F2, and F4 when $\lambda = 60\%$ and achieved the best the best results on problem F3 when $\lambda = 10\%$. We believe that the optimal setting for the λ is largely attributed to the characteristics of the given problem. However, for simplicity, we set $\lambda = 60\%$ because it allows sGP to perform reasonably well on all problems. In future research, we will explore methods to allow the setting of the λ automatically.

5. Conclusions

In this work we propose a new form of GP called sGP. The proposed framework evolves μ percentage of the population using standard search operators and the remaining λ percentage using surrogate models. For the λ part of the population, sGP uses surrogate model as a search operator to produce new offspring. sGP uses two specialised models, namely, semantic-model to map the syntactic space of the problem into semantic space and fitness-model to map the semantic space into fitness space. The contribution of this paper is fourfold; 1) To the best of our knowledge, this is the

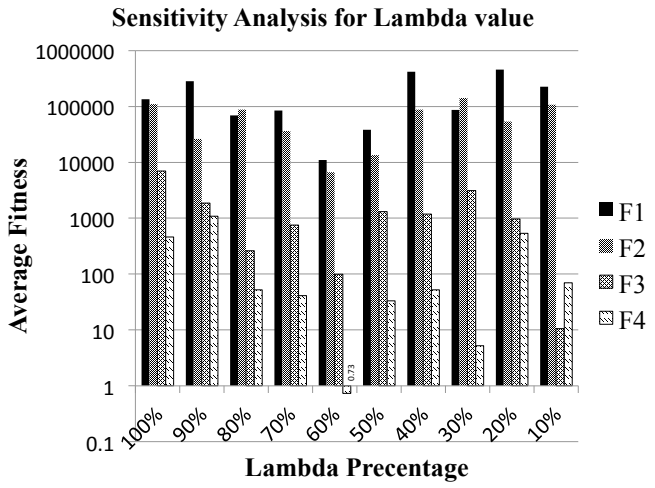

Figure 5: sGP performance under different value of λ. Results are averaged from 20 different runs.

first successful attempt to apply surrogate model to GP tree-like representation, 2) We extend semantic operators to computationally expensive problems, 3) The proposed framework can be generalised, in principle, and it is not problem-dependant because surrogate approximates the semantic search space instead of the structural space. This allows surrogate to use the natural metric distance of the semantics representation, and 4) A novel form of GP is presented, which we called sGP.

Although, sGP adds a layer over a standard GP system, this increased complexity is justified by the better performance which is demonstrated by the experimentations when all algorithms included in the comparison tested under the exact same number of evaluations. Empirical evidence of the new framework with three different problems (Symbolic regression, Even n-parity, and Time-series forecasting) demonstrate its superiority over standard GP systems.

It is fair to report that the main disadvantage of sGP is that it requires the use of original problem fitness measure in order to create training examples for the semantic-model to predict trees' semantic. This part of surrogate training comes at extra computational cost. However, we used only small numbers of samples from the training set and thus a significant reduction of the full computational costs (coming from evaluating trees on the full training set in standard semantic aware operators) is still achieved. In addition, we compared our system against different GP systems after allocating them exactly the same number of fitness evaluations and results were in favour of sGP. Also, another disadvantage is that the surrogate model may only be applied on problems with specific search space properties. In particular, each individual should produce one and only one vector value for each input. Assuming a given vector of inputs will produce corresponding vector of outputs. This restriction excludes problems where the GP trees are used directly as programs, like Artificial Ant, or where the produced output is not numeric or vector-like.

For future research, we will try to avoid the extra computational costs required to train the semantic-model. In addition, we will apply sGP on real-world computationally expensive problems where semantic calculations can not be easily obtained (e.g., robot vision). Moreover, we will look closely to the surrogate behaviour with regard to different problems' space. Also, for future work, we will explore the framework performance with different surrogate models other than RBFNs.

References

[1] E. Acar and M. Rais-Rohani. Ensemble of metamodels with optimized weight factors. *Structural and Multidisciplinary Optimization*, 37(3):279–294, 2009.

[2] J. R. Anderson, R. S. Michalski, R. S. Michalski, T. M. Mitchell, et al. *Machine learning: An artificial intelligence approach*, volume 2. Morgan Kaufmann, 1986.

[3] L. Beadle and C. G. Johnson. Semantically driven crossover in genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 111–116. IEEE, 2008.

[4] L. Beadle and C. G. Johnson. Semantically driven mutation in genetic programming. In *IEEE Congress on Evolutionary Computation*, pages 1336–1342. IEEE, 2009.

[5] A. G. Bors. Introduction of the radial basis function (rbf) networks. Technical report, Department of Computer Science, University of York, UK, 2001.

[6] N. L. Cramer. A representation for the adaptive generation of simple sequential programs. In J. J. Grefenstette, editor, *Proceedings of an International Conference on Genetic Algorithms and the Applications*, pages 183–187, Carnegie-Mellon University, Pittsburgh, PA, USA, 24-26 July 1985.

[7] D. L. Donoho. High-dimensional data analysis: the curses and blessings of dimensionality. *American Mathematical Society Conference on Math Challenges of the 21st Century*, August 2000.

[8] M. Emmerich, A. Giotis, M. Özdemir, T. Bäck, and K. Giannakoglou. Metamodel-Assisted Evolution Strategies. *Parallel Problem Solving from NaturePPSN VII*, 2439:361–370, 2002.

[9] K. C. Giannakoglou. Design of optimal aerodynamic shapes using stochastic optimization methods and computational intelligence. *Progress in Aerospace Sciences*, 38(1):43–76, 2002.

[10] T. Goel, R. T. Haftka, W. Shyy, and N. V. Queipo. Ensemble of surrogates. *Structural and Multidisciplinary Optimization*, 33(3):199–216, 2007.

[11] D. Jackson. Promoting phenotypic diversity in genetic programming. In *Proceedings of the 11th international conference on Parallel problem solving from nature: Part II*, PPSN'10, pages 472–481, Berlin, Heidelberg, 2010. Springer-Verlag.

[12] Y. Jin. A comprehensive survey of fitness approximation in evolutionary computation. *Soft Comput.*, 9(1):3–12, 2005.

[13] A. Kattan and E. G. Lopez. Evolving radial basis function networks via gp for estimating fitness values using surrogate models. In *IEEE Congress on Evolutionary Computation*, pages 1–7. IEEE, 2012.

[14] A. Kattan and R. Poli. Genetic programming as a predictor of data compression saving. In *Evolution Artificielle, 9th International Conference*.

[15] A. Keane and N. Petruzzelli. Aircraft wing design using GA-based multi-level strategies. *AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, Long Beach, USA 06 - 08 Sep 2000.*, 2000.

[16] J. R. Koza. *Genetic Programming II: Automatic Discovery of Reusable Programs*. MIT Press, Cambridge Massachusetts, May 1994.

[17] F. H. Lesh. Muti-dimensional least-squares polynomial curve fitting. *Communications of ACM*, 2(9):29–30, 1959.

[18] D. Lim, Y. Jin, Y.-S. Ong, and B. Sendhoff. Generalizing surrogate-assisted evolutionary computation. *Evolutionary Computation, IEEE Transactions on*, 14(3):329 –355, 2010.

[19] D. Lim, Y. S. Ong, Y. Jin, and B. Sendhoff. A Study on Metamodelling Techniques, Ensembles, and Muti-Surrogates in Evolutionary Computation. *GECCO '07: Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 1288–1295, 2007.

[20] D. Lim, Y. S. Ong, Y. Jin, and B. Sendhoff. Evolutionary Optimization with Dynamic Fidelity Computational Models. *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, pages 235–242, 2008.

[21] D. J. Mackay. Introduction to Gaussian processes. *Neural Networks and Machine Learning*, 168:133–165, 1998.

[22] M. C. Mackey and L. Glass. Oscillation and chaos in physiological control systems. *Science*, 197:287, 1977.

[23] A. Moraglio. Geometric unification of evolutionary algorithms. In *European Graduate Student Workshop on Evolutionary Computation*, page 45, 2006.

[24] A. Moraglio and A. Kattan. Geometric surrogate model based optimisation for genetic programming: Initial experiments. Technical report, University of Birmingham, 2011.

[25] A. Moraglio, K. Krawiec, and C. G. Johnson. Geometric semantic genetic programming. In *Parallel Problem Solving from Nature, PPSN XII (part 1)*, 2012.

[26] Q. U. Nguyen, X. H. Nguyen, and M. O'Neill. Semantics based mutation in genetic programming: The case for real-valued symbolic regression. In *15th International Conference on Soft Computing, Mendel'09*, 2009.

[27] Y. S. Ong, P. Nair, A. Keane, and K. Wong. Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. *Knowledge Incorporation in Evolutionary Computation, Studies in Fuzziness and Soft Computing Series. Springer Verlag*, 2004.

[28] Y. S. Ong, P. B. Nair, and A. J. Keane. Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modelling. *American Institute of Aeronautics and Astronautics Journal*, 41(4):687–696, 2003.

[29] R. Poli, W. B. Langdon, and N. F. McPhee. *A Field Guide to Genetic Programming*. Published via `http://lulu.com`, 2008. (With contributions by J. R. Koza).

[30] M. J. D. Powell. Radial basis functions for multivariable interpolation: a review. pages 143–167, 1987.

[31] Y. Tenne. A Model-Assisted Memetic Algorithm for Expensive Optimization Problems. *Nature-Inspired Algorithms for Optimisation*, pages 133–169, 2009.

[32] H. Ulmer, F. Streichert, and A. Zell. Evolution strategies assisted by Gaussian processes with improved preselection criterion. *The 2003 Congress on Evolutionary Computation, 2003. CEC '03*, 1:692–699, 2004.

[33] N. Q. Uy, N. X. Hoai, and M. O'Neill. Semantic aware crossover for genetic programming: The case for real-valued function regression. In *EuroGP*.

[34] N. Q. Uy, M. O'Neill, N. X. Hoai, B. Mckay, and E. Galván-López. Semantic similarity based crossover in gp: the case for real-valued function regression. In *Proceedings of the 9th international conference on Artificial evolution*, EA'09, pages 170–181, Berlin, Heidelberg, 2010. Springer-Verlag.

[35] V. N. Vapnik. *Statistical Learning Theory*. Wiley New York, 1998.

[36] M. J. Zaki and M. Sayed. The use of genetic programming for adaptive text compression. *Int. J. Inf. Coding Theory*, 1(1):88–108, 2009.

[37] L. E. Zerpa, N. V. Queipo, S. Pintos, and J.-L. Salager. An optimization methodology of alkaline-surfactant-polymer flooding processes using field scale numerical simulation and multiple surrogates. *Journal of Petroleum Science and Engineering*, 47(3-4):197–208, 2005.

[38] Z. Zhou, Y. S. Ong, P. B. Nair, A. J. Keane, and K. Y. Lum. Combining global and local surrogate models to accelerate evolutionary optimization. *IEEE Transactions on Systems, Man and Cybernetics (SMC), part C*, 37(1):66–76, 2005.

[39] Z. Z. Zhou, Y. S. Ong, M. H. Lim, and B. S. Lee. Memetic algorithm using multi-surrogates for computationally expensive optimization problems. *Soft Computing Journal*, 11(11):957–971, 2007.

[40] J. Zurada. *Introduction to Artificial Neural Systems*. West Publishing Co., St. Paul, MN, USA, 1992.